ampus
omputing
etwork

"A TOOL-BEARING HOST IN THE NATIONAL SOFTWARE WORKS"

Semiannual Technical Report
December 1975 - June 1976

CCN/TR10

R. T. Braden

H. C. Ludlam

niversity of
alifornia,
os
ngeles

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) CCN/TR10	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) A Tool-Bearing Host in the National Software Works.		5. TYPE OF REPORT & PERIOD COVERED Semi-annual Technical Report 12/1/75 - 6/30/76
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) R. T. Braden, H. C. Ludlam		8. CONTRACT OR GRANT NUMBER(s) (15) MDA 903-74-C-0083
9. PERFORMING ORGANIZATION NAME AND ADDRESS Campus Computing Network, UCLA C0012, Los Angeles, California 90024		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Code 4P10, ARPA Order No. 2543/3
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency, 1400 Wilson Blvd, Arlington, Virginia 22209		12. REPORT DATE (11) 1 April 1977
		13. NUMBER OF PAGES 68
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (9) Semi-annual technical rept. 1 Dec 75-30 Jun 76		15. SECURITY CLASS. (of this report) None
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution unlimited. (12) 71p.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

National Software Works, batch processing, RJE protocols, performance measurement, reliability, failure recovery, batch tools, tool installation, file attributes, file conversion.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the technical development relating to the National Software Works (NSW) at UCLA-CCN during the period December 1, 1975 through June 30, 1976. The NSW is a distributed multi-computer operating system based on the ARPANET. The primary goal of the NSW project at CCN is to make the CCN 360/91 a "tool-bearing host" within the NSW.

over

(continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

410 283

LB

Section 2 of this report describes the further development of the batch job facility for the NSW. Section 3 deals with the installation of batch tools at CCN. Section 4 describes research relating to the design of the file-movement mechanism for NSW, the File Package.

CAMPUS COMPUTING NETWORK

University of California at Los Angeles
405 Hilgard Avenue,
Los Angeles, California 90024

"A TOOL-BEARING HOST IN THE NATIONAL SOFTWARE WORKS"

Semiannual Technical Report
December 1975 - June 1976

CCN/TR10

R. T. Braden

H. C. Ludlam

A

This work was sponsored by the Advanced Research Projects Agency
of the Department of Defense, under Contract Number
MDA903 74C 0083, Order 2543/3.

REPORT SUMMARY

This report describes the technical development relating to the National Software Works (NSW) at UCLA-CCN during the period December 1, 1975 through June 30, 1976. The NSW is a distributed multi-computer operating system based on the ARPANET. The primary goal of the NSW project at CCN is to make the CCN 360/91 a "tool-bearing host" within the NSW.

NSW work at CCN falls generally into three areas: (1) implementation of a batch remote job entry facility for NSW, the IP Server; (2) implementation of facilities to support interactive tools under the TSO timesharing system; and (3) installation of tools on the CCN system.

Section 2 of this report describes the further development of the batch job facility for the NSW. Section 3 deals with the installation of batch tools at CCN. Section 4 describes research relating to the design of the file-movement mechanism for NSW, the File Package.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. THE NATIONAL SOFTWARE WORKS

The ARPANET is the prototypical large-scale resource-sharing network of diverse host computers. Creation of the ARPANET required two distinct development phases: first, invention of a new communication technology, "packet switching", and its realization in the communication subnet of IMP's and TIP's; and second, development of inter-host protocols for remote terminal access, file transfer, remote job entry, etc.

The ARPANET allows a sophisticated programmer to use a large number of powerful programming tools on many remote hosts, without regard to their geographical distance. From the viewpoint of an ARPANET user, the diversity of hosts and operating systems on the Network is at once the greatest advantage of Network access, and also the greatest barrier to effective Network use. In particular, a user must have an account and be knowledgable about the command language and file system on every host that he wishes to access.

The National Software Works, or "NSW", development is attempting to overcome this diversity barrier by effectively building a distributed multi-computer operating system spanning a set of server hosts [1]. In particular, the NSW will offer a user:

* a single global file system spanning all the NSW server hosts;

* a single account, with the NSW;

* a standardized interaction flavor across all interactive systems, e.g., standard ways to ask for help and to signal "Attention" or "Break"; and

* a single interactive job control language generator for all batch jobs.

Thus, a user will log into the NSW and then may execute programs, called "tools", by name. He does not need to know which server host is actually executing a particular tool. When the tool requires a file, the user enters the NSW file name. The NSW file mechanism automatically locates a suitable physical copy of that file, and if necessary copies it across the ARPANET into the local file-space in which the tool is running. One may consider that the NSW represents the third major phase of ARPANET development.

The NSW is designed to provide these facilities without requiring significant changes to the operating systems of the participating hosts. The objectives of the NSW, which seem very grand, are in fact strictly limited. The NSW does NOT require rewriting the operating systems of the participating hosts, nor does it require standardization of Fortran compilers or text editors, for example. Only the file system, the accounting, and certain surface features of the command language are to be standarized.

The central controlling and record-keeping mechanism of NSW is the Works Manager, or "WM" [2]. The Works Manager can be likened to an operating system for the server hosts included in the NSW. For example, the WM performs the following operating system functions:

* controlling access to programs and files;

* allocating NSW system resources;

* maintaining the NSW global file catalog; and

* accumulating cost and resource-usage measures.

The WM keeps a data base in which are stored the user profiles and access rights, the file catalog, and the tool descriptors. Functionally, the WM is organized into a set of subroutines or "procedures" which are "called" by the other components of NSW. In a sense, the WM procedures are like the Supervisor Call routines normally included in an operating system; each procedure is called synchronously to perform a single well-defined function. As a side effect of these calls, the WM builds and maintains permanent information in its data base as well as dynamic tables of the currently-active users and tools.

1.2. THE NSW AT UCLA-CCN

In the terminology of the NSW, a server host is called a "Tool-Bearing Host", or "TBH". The work described in this Technical Report is aimed at making the IBM 360/91 at UCLA-CCN a Tool-Bearing Host for the NSW.

Initial TBH software development at CCN resulted in the implementation of a facility to execute "batch tools", i.e., tools which can be executed in background using the normal batch-processing machinery of the IBM operating system OS/MVT. The principal software component required at CCN for this purpose was a server for the "IP" protocol. The history and definition of IP and the implementation of the CCN server were fully discussed in a previous CCN Technical Report [3]. Section 2 of the present report describes further development of the NSW batch job system at CCN.

Section 3 discusses the installation of batch tools, with particular reference to CCN.

During the period covered in this report, work began at CCN on the software components necessary for interactive tools under TSO. The necessary components are:

* An MSG Server

  MSG is the interprocess communication protocol for NSW [4]. MSG is oriented towards messages rather than connections, and thus each message carries the destination process address.

* A File Package ("FP")

  The File Package is invoked directly by the Works Manager or indirectly by another File Package to perform NSW file operations -- copying files locally or across the ARPANET, converting their representation to/from a form suitable for a particular tool, and deleting files [5].

* A Foreman ("FM")

  The Foreman is logically the extension of the local operating system required for NSW; it is also the interface between the Works Manager and the tools [6].

Design of the File Package (like its predecessor FTP) is difficult because complex conversions of data representation are often required when files are transferred between different CPU's and operating systems [Note 1]. NSW is designed with the assumption that these conversions can be performed automatically as a file is moved from one host to another. In order to accomplish automatic file conversion in this manner, the File Package design must choose the attributes to be stored for each file. Section 4 of this Report contains an analysis of file attributes and their impact on the File Package specifications.

2. BATCH TOOL-BEARING HOST IMPLEMENTATION

Under NSW, a "batch tool" is defined simply as a tool which is executed in the background, i.e., without an online terminal attached [6]. Any TBH which is capable of executing a batch tool could be called a "batch TBH", or "BTBH". The actual mechanism for executing the batch tool will depend upon the BTBH operating system. For example, some purely interactive systems (e.g., the PDP-10 TENEX system) execute "batch" jobs as special cases of interactive jobs, simply replacing terminal I/O by disk file I/O. At the other extreme are the traditional batch-processing systems which queue jobs for exclusive access to one of a small number of multiprogrammed virtual CPU's. For efficiency, these batch-processing systems generally require that input files for batch tools be "pre-staged". That is, before a batch job can be added to a batch-processing queue ("submitted"), all of the input files must be made resident in that host's local file-space.

The IBM operating system OS/MVT used on CCN's 360/91 includes both traditional batch-processing and an interactive timesharing system, TSO. The two are essentially compatible, but distinct. An interactive job under TSO must have a terminal attached; a batch tool must be executed in the normal batch-processing system of OS/MVT.

To execute any batch tool under the NSW, a user must interact with a program in the Works Manager called the Interactive Batch Specifier, or "IBS". The IBS queries the user for file names, resource limits, and other parameters of the job, and builds job description tables. When the job description tables are complete the IBS passes them to the WM and exits, allowing the user to enter another interactive program while the WM runs the job in the background. The user can later query the WM for the status of his jobs, or the WM may send him a message when output from a job is available for viewing at his console.

Batch job execution under the NSW is actually controlled by a WM process called the Works Manager Operator, or "WMO". When a new batch job is specified by a user, the WM passes the description of the job to the WMO, which maintains tables of all batch jobs currently in progress. The WMO invokes primitive operations at the BTBH's to cause these jobs to be processed.

2.1. BATCH JOB MODELS

A vital design issue for the batch job mechanism for the NSW is: to what extent should the control functions be centralized in the WMO instead of decentralized to the BTBH's? This report will discuss briefly the following three models for batch job control, listed in order of decreasing centralization of functions:

a. IF Server Model

b. Batch Job Package ("BJP") Model

c. Batch Foreman ("BF") Model

We begin withthe BJP model. Let us postulate that the WMO has responsibility for sequencing batch jobs through each stage of processing -- pre-staging input files, submitting the JCL file, polling or waiting for notification of job completion, and retrieving output files. The primitive operations required at each BTBH can be divided into file operations and job-related operations. Using the standard NSW mechanisms, these primitives would be invoked at the BTBH via the MSG interprocess communication facility [4], and the file operations would naturally be collected into a File Package ("FP") [5]. We then postulate a "Batch Job Package" for the job-related functions.

Specifically, the BTBH functions required by the WMO are:

File Package Functions --

* Fetch an input file from a remote host on the Network, translating it into the format required by the batch tool.

* Make a copy of a local file, translating it into the format required by a batch tool.

* Send an output file over the Network.

* Delete a file from the local BTBH file-space.

Batch Job Package Functions--

* Submit a Job, i.e., cause a specified local file to be interpreted as a job-control stream by the BTBH operating system.

* Delete a batch job from the tables and queues of the batch system.

* Spontaneously notify the WM when batch job output is ready for retrieval.

* Answer queries from the Works Manager on batch job status.

The BJP model of batch job submission requires that an MSG server, a File Package, and a BJP all be written for each BTBH (or at least each family of equivalent BTBH's). It is anticipated that some hosts which provide only batch processing to the NSW will be too small or simple to support all of this machinery. For such "mini-hosts", the IP Server model had been developed to minimize the new software at the BTBH.

Under the IP Server model, the BTBH interface software for the NSW consists of a server process for the "IP" protocol. The IP Server is purely a slave of the WMO, which fully controls batch execution. Therefore, IP is basically a demand/response protocol; WMO sends IP messages to the BTBH to invoke the batch-submission and file-movement primitives, and the IP Server returns success/failure replies to WMO.

IP allows WMO to invoke any of the primitives listed above, except local copy. Local copy was omitted from the simplified protocol since real batch tools seldom clobber their input files (and OS/MVT, for example, will enforce read-only access to input files.) The basic IP protocol was designed by another contractor, Massachusetts Computer Associates (MCA). MCA also designed and implemented the IBS, WMO, and IP User programs to execute on a PDP-10 under the TENEX operating system.

Although the CCN 360/91 system is not a mini-host, it was simple and expedient to implement an interim BTBH service at CCN as a server program for IP. This implementation was described in the previous CCN Semiannual Technical Report [3]. The IP Server at CCN is implemented as a TSO command processor named "MP". MP is written almost entirely in PL/1 and is executed by a "pseudo-user", i.e., as a system job under TSO.

In the original IP Server implementation at CCN, the server job is permanently logged onto TSO, using a virtual terminal controlled by a system process within the NCP. This same NCP process, called "IPTASK", also (1) listens on a pair of ARPANET sockets for Request-for-Connections from the WMO, and (2) opens an internal interprocess communication path to MP, using the "Exchange" [7]. After the WMO does open the Network connections, a subprocess of IPTASK simply relays IP messages between the Network connection-pair and the Exchange window to MP. Thus, the Exchange calls within MP essentially form a system-call interface for the ARPANET connections to WMO. The diagram of Figure 1 illustrates this organization.

Under the IP Server and BJP models, the WMO centralizes those functions which are characteristic of an interactive tool Foreman [6]: interfacing between the TBH operating system and the Works Manager. For example, the WMO would issue the WM procedure calls [2] to GET the input files and to DELIVER the output files to the NSW file system.

The advantages of this centralization of function are clear: there will presumably be (many) more different BTBH implementations than WM implementations in the ultimate NSW. On the other hand, the design of NSW batch may be simplified conceptually if batch tool execution is handled in closer analogy with interactive tools. This leads to a model based upon a "Batch Foreman" (BF) on each BTBH.

The "Batch Foreman" is like an interactive Foreman in serving as the local (to the BTBH) interface between the operating system and the WM. Once assigned a batch job by the WMO, the BF would take responsibility for "driving" the local operating system to execute that job. In particular, the BF, like an interactive tool Foreman, would issue the file GET and DELIVER calls to the Works Manager. This would require that WMO pass the IBS job description tables and JCL skeleton to the BF for interpretation.

The BF would not be perfectly analogous to an interactive Foreman, however. The natural model for interactive tools uses a Foreman process per interactive tool session; however, it seems conceptually simpler to have a single BF process per BTBH rather than one per job awaiting execution.

Figure 1 : Network IP Server Implemented on 360/91

```
        T E N E X                           U C L A / C C N
       P D P - 1 0                            3 6 0 / 9 1
                                    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxx          x                               x
x                        x          x                               x
x    "WMO"               x          x "IPTASK"            "MP"      x
x                        x          x                               x
x :----------:           x          x                   :---------: x
x :          : :-------: x   <IP>   x  :-------:  <IP>  :         : x
x : WORKS    : :       : x          x  :       :        :  I P    : x
x : MANAGER  : :  I P  : <-------------:Network:<-------:         : x
x : OPERATOR :-: USER  :-------------->:inter- :------->: SERVER  : x
x :          : :PROGRAM: x ARPANET  x  :  face :Exchange:         : x
x :          : :       : x          x  :       :        :         : x
x :          : :-------: x          x  :-------:        :         : x
x :----------:           x          x  NCP    .         :---------: x
x                        x          x         .........>:  TCAM   : x
x                        x          x     (pseudo-user) :---------: x
xxxxxxxxxxxxxxxxxxxxxxxxxx          x                    TSO Job    x
                                    x                               x
                                    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

2.2. IP SERVER DEVELOPMENT

During the period reported on here, a number of changes to the original IP Server implementation were planned and executed by CCN. These changes were generally designed to improve the performance, reliability, and maintainability of the implementation.

2.2.1. MULTIPLE IP SERVERS

The original definition of the IP protocol used the simplest possible scheme for Network connections between the WMO and the various IP Server processes. It was assumed that there would be only one WMO process in the NSW (since there is only one Works Manager), and in fact only one WMO on the ARPANET. Each IP Server host was to have a fixed "known" pair of sockets to which the WMO would open a full-duplex connection pair. At CCN, the fixed pair of sockets (258, 259) was chosen to avoid conflict with ICP sockets and other CCN socket allocations.

The IP Server is always passive and only "listens" on its fixed sockets; the IP User process (i.e. the WMO) always initiates the connections by sending RFC's. If the Server does not respond, the User process must time out, wait a reasonable interval, and try again.

The WMO was designed by MCA to close the connections to an IP Server host when there are no jobs for that host in the WMO queues. However, the TSO session for MP remains logged on even after the Network connections close. MP is swapped out by TSO, pending a wakeup signal when a new IP message arrives through the open Exchange window.

If MP should abnormally terminate, IPTASK will restart it after a short delay. In the meantime, however, the Network connections are closed by CCN.

This simple model quickly proved inadequate for the developing requirements of the NSW. It was decided to set up two NSW systems, each with its own WMO. The production NSW was to be on host ISIC and a test system on host BBNB. Since the IP Server was not designed to multiplex multiple WMO's, the following scheme was adopted:

* A separate IP Server process is installed at CCN for each WMO host. Each server process is realized by a distinct TSO session (job) that is permanently logged on and executing the MP module.

* Each server session executes as a different pseudo-user in its own "logon directory" [Note 2]. As a result, each IP Server process is parameterized by its own private configuration file [8]. In particular, each configuration file specifies a different account number for submitting batch jobs, giving each server process a private name-space for batch jobnames (see below). Furthermore, each configuration file specifies a different notification queue (used by OS/MVT for signaling completion of batch jobs.)

* There is a separate IPTASK process for each WMO host, listening for RFC's from that host on the same fixed sockets. To simplify maintenance of this multi-host system, the IPTASK code was made table-driven; the table contains a list of WMO hosts and corresponding pseudo-user logon directories (userid, account) for the IP Server sessions.

* In order to connect each IPTASK to the corresponding MP incarnation, both MP and IPTASK normally use the userid of the MP session as the symbolic tag ("MYTAG" for MP, "YOURTAG" for IPTASK) for opening the Exchange window.

The NCP at CCN tries to enforce strictly the rules of the ARPANET host-host protocol, including the rule that a particular local socket on a host can participate in only one Network connection. It is this uniqueness of socket usage that forces the reconnection implicit in the ICP sequence, and distinguishes the host-host protocol from the proposed internetwork TCP protocol [9], for example.

This socket restriction would have made it impossible to have connections to both BBNB and ISIC simultaneously from CCN sockets (258,259). It is interesting, as well as fortunate, that this restriction could be removed by merely excising the uniqueness-checking code from the CCN NCP. The data structures in the NCP were adequate to allow connections to the same CCN socket from DIFFERENT FOREIGN HOSTS, but not different foreign sockets on the same host. The CCN NCP was changed in this manner to allow WMO connections from multiple hosts to the same fixed CCN sockets.

2.2.2. IP FOR FILE TRANSFER

The fixed-socket model for Network connections again became inadequate when it was decided to use the IP Server as an interim file transfer mechanism, pending the design and implementation of the NSW File Package at CCN. In the original IP design, all input files were to be pre-staged by WMO to the batch TBH, and after job completion all output files were to be copied back over the Network to the WMO host. Initial experience with installing actual user tools (see Section 3 below) showed that tools often produce multiple output files, but that the user often looks at only one of them (usually, an error file). Moving all the output files unnecessarily wastes valuable Network transfer time and delays other IP jobs.

A better scheme would be to set up the batch job descriptors to automatically return only those output files which the user commonly wants to see, e.g., error-report files. Other NSW files produced by the job would be left in NSW file-space at the BTBH and be cataloged in ("delivered to") NSW. These files could be fetched across the Network later on demand of some other tool or deleted by the user.

Normal NSW file transfer operations are performed by the File Package ("FP"). The Works Manager calls the File Package on the target host, which in turn calls the FP on the source host and arranges to transfer the file. As an interim, the TENEX FP was modified by MCA to recognize an IP Server host and to use the IP file transfer mechanism in that case.

So far, this seems to be a simple if pragmatic scheme which should be transparent to the batch TBH. However, MCA found implementation problems within the TENEX NSW system; it turned out to be much easier for the File Package to invoke the IP User program directly under the FP process(es) rather than to queue requests for the single existing WMO process. This meant proliferation of IP User processes on the same host, a situation which could not be handled with fixed Network sockets at CCN.

It was therefore decided to allow invocation of the IP Server at CCN via a normal ICP (Initial Connection Protocol) sequence.

* MCA modified the IP User program under TENEX to initiate an ICP to connect to CCN, when called either from the WMO or from a File Package process incarnation.

* ICP sockets 199 and 197 were assigned respectively to the "Jobs" servers (invoked by WMO), and to the "File Transfer" servers (invoked by the TENEX File Package).

* The IPTASK program within the CCN NCP was modified to create a new data-relay subprocess whenever an ICP to either of these sockets occurs.

* Each such IPTASK subprocess selects an idle NSW logon directory (see Note 2), logs onto TSO as a pseudo-user, and issues the "MP" command to execute the server module.

* When the Network connection closes, the IP Server job in TSO is logged off, and the IPTASK subprocess in the NCP vanishes.

The result is to create a variable but bounded number of IP Server processes as concurrent TSO jobs. To prevent confusion of the batch jobs submitted by these servers, it is necessary to restrict the number of concurrent "Jobs" servers (used by WMO for batch job entry operations) to a maximum of one per WMO host. Since there is never supposed to be more than one WMO per WM host, this restriction should never be tested. To avoid excessive use of CCN resources, the number of simultaneous "File Transfer" server processes was also limited to a (small) integer N.

Thus, at any time there can be at most W "Jobs" servers and N "File Transfer" servers active at CCN (where W is the number of distinct WMO hosts, currently 2.) If an ICP in excess of either limit is attempted, CCN will queue the RFC until an existing session terminates. We considered sending a canned "server ports busy" message in this case, but the programming cost exceeded the expected benefit.

In summary, the CCN IP Server can now be invoked in three ways:

* by connections (initiated by WMO) directly to the fixed sockets (258,259);

* by ICP (from the WMO) to socket 199; and

* by ICP (from an instance of the TENEX FP) to socket 197.

The following Table 1 shows the current configuration of IP Servers installed at CCN, as an example.

Table 1 : Current IP Configuration at CCN

WMO Host	CCN Socket	Server Type	TSO Directry (acct.userid)	Batch Account	Exch. Tag	GMF Queue
BBNB	258,259	old	AHA179.NW1	NSW011	NW1	NW1NOTE
ISIC	258,259	old	AHA183.NW2	NSW001	NW2	NW2NOTE
BBNB	ICP->199	"Jobs"	AHA179.NWx	NSW011	NWx	BBNNOTE
BBNB	ICP->197	"FileT"	AHA179.NWy	------	NWy	(null Q)
ISIC	ICP->199	"Jobs"	AHA183.NWx	NSW001	NWx	ISINOTE
ISIC	ICP->197	"FileT"	AHA183.NWy	------	NWy	(null Q)

Here:
x= one of '5', '6', or '7'.
y= one of '8', '9', 'A', 'B', or 'C'.

2.2.3. FULLY-QUALIFIED FILE NAMES

Before sending an input file to the batch TBH, WMO sends an IP ENTER FILE NAME request to select and return the local file name under which the BTBH will store the file. The first version of the IP Server at CCN always returned an unqualified (i.e., a directory-relative) file name [Note 2] to the WMO. Unfortunately, the OS/MVT Job Control Language ("JCL") requires that the actual working directory be named explicitly. Thus, the JCL kernel used by the IBS contained the CCN directory in which the job was to run.

This approach became untenable for two different reasons. Firstly, the JCL kernel is actually compiled into the IBS code [Note 3]. Hence every administrative change of configuration at CCN required a recompilation of the TENEX IBS program. Secondly, the multiple IP Servers discussed above required multiple directories.

To meet these objections, we changed the IP Server to return to WMO a "fully-qualified" file name, i.e., a name string which includes the working directory for the batch jobs it submits. The IP Server uses the TSO rules, in which a fully-qualified file name (i.e., DSNAME) is enclosed in quotes. Such a quoted name is acceptable in JCL.

2.2.4. PERFORMANCE IMPROVEMENT

Two significant extensions to the IP Protocol were developed by CCN and MCA to improve the ARPANET transmission efficiency of IP file transfers.

* Record Blocking

  Multiple IP records were blocked into a single IP message. The maximum message size was set at 1K bytes for both the User (PDP-10) and Server (360/91) programs.

* Data Compression

  To further improve Network bandwidth utilization, CCN and MCA decided to implement in IP the data compression scheme designed by MCA for the File Package Intermediate Language or "IL" [5].

The resulting changes to the IP Protocol will be documented in a later CCN technical report.

Upon completion of an IP file transfer operation, the CCN server writes a statistical line into the history log. This line includes two measures of transmission rate, "Real Baud" and "Effective Baud". The first is the number of bits transmitted over the ARPANET divided by the elapsed time; this includes the IP headers. The second is the number of data bits written/read from the CCN disk file, again divided by the elapsed time. We have examined the ratio of these two Baud rates as a measure of the effectiveness of the data compression.

Note that the original IP User and Server programs did truncate trailing fill characters (blanks for character data, zero for binary) before transmitting over the Network. The IL encoding also provides "internal" compression: encoding strings of replicated single characters into counts, with particular attention to the fill character. In addition, IL encodes sequences of "empty" lines (containing only fill characters) in an extremely compact format. However, for those files which are recorded on disk at CCN with fixed-length records, truncation may be more significant than internal compression in the total compression ratio.

The last column of the following table shows some typical compression ratios (Real-Baud divided by Effective-Baud).

Table 2 : Typical Data Compression Statistics -- IP Server

OPERATION	SIZE	CONTENTS	DISK FORMAT(*)	COMPRESSION RATIO
Receive	97 Lines	CMS2M Source	FB,80	43%
Receive	202 Lines	Fortran Source	FB,80	33%
Receive	878 Lines	MACRO20 Source	FB,80	19%
Send	146 Lines	JCL Listing	VBA	47%
Send	562 Lines	CMS2M Listing	VBA(**)	30%
Send	1034 Lines	MACRO20 Listing	VBA(**)	31%
Send	18 Cards	CMS2M Object Deck	FB,80	47%
Send	55 Cards	MACRO20 Object	FB,80	86%

NOTE * :

Disk Format = "FB,80": records are written on disk with a fixed length of 80 bytes. However, "VBA" records are written as variable-length records and can generally be expected (as in the JCL listing file) to have trailing blanks already deleted on disk. However, see the following note:

Note ** :

Although these output files are stored as variable-length records, the compiler implementations always write listing records which are padded with blanks to 120 characters. Thus, the output is effectively "FB,120".

The first three cases in Table 2 illustrate the variation in compression ratio with the "density" of different source languages. The particular MACRO20 program was almost free of comments, providing opportunity for 5:1 compression (including both truncation and internal compression.) The fourth case and the last case show examples of purely internal compression; the last case is interesting as it is a pure "binary deck".

In addition to the external improvements discussed above, several internal performance improvements were made within MP. For example, non-data-transfer IP messages are now blocked as much as possible into the same 1K buffer before being sent via Exchange to IPTASK and the Network.

Specifically, when a new message has been assembled, the last buffer on the output queue is examined to determine whether it has room for the new message. A new buffer is obtained and enqueued only if the message will not fit into the previous buffer.

A better understanding of the storage management algorithms used by the IBM PL/1 Optimizing Compiler allowed an increase of the number and size of buffers within MP.

2.2.5. INSTRUMENTATION

To aid in testing new versions of either the IP server or the WMO, an online monitoring facility was added to MP. New TSO logon "directories" [Note 2] were established for this purpose, one for each WMO host. If a person logs onto TSO using one of these directories, he will receive on his terminal an online copy of the history logs kept by all active MP processes for that host. This useful feature was implemented by simply adding "TPUT tjid" calls to the monitoring subroutine of MP (see [3]); this terminal output command sends each log line to the terminal with the monitoring userid. If no one is logged on at the time, the system simply turns the TPUT into a "NOP" with an appropriate return code.

A complete description of the procedure for using this logging mechanism and interpreting the output is contained in Appendix B.

This very elementary mechanism has been extremely useful in bringing up new versions of the NSW and in testing new versions of the IP server. Note that one of the configuration parameters in each MP process determines whether or not the log is to include each record of IP file information moving in or out of CCN.

2.3. PERFORMANCE MEASUREMENTS

The IP Server module MP at CCN writes into its history log some crude measurements of file transfer performance. Examination of the data logged from a variety of NSW batch jobs at CCN gives a rough idea of performance. However, it also reveals serious defects in the current instrumentation of the IP Server.

We will now present a typical sample of the data that can be collected at present, and show the (limited) conclusions that can be drawn. The following table shows six IP SEND operations which occurred in sequence over two days. All files were Fortran source decks being pre-staged to CCN through NSW (in fact, cases B-F were apparently the same source program.)

Cases A-E form an interesting sequence since they span a time period when the load at CCN was gradually increasing from idle. This appears in the increasing count of TSO swaps of the MP job and in the gradual increase in CPU usage (caused by a defect in OS/MVT CPU timing that charges active tasks for interrupt service overhead.)

The most interesting statistic in Table 3 is REAL BAUD, which is the average number of bits per second received over the ARPANET. This figure is seen to vary dramatically over nearly a 5:1 range; furthermore, this variation seems to be fairly typical. Observe that in cases A and B the TSO swap count is 0; that is, MP was never swapped out, but rather was resident in core memory during the entire file transfer. The disk I/O at CCN (5 block writes) could not account for more than a few seconds of the observed elapsed time, even in rare circumstances.

On the other hand, we believe that the variation in REAL BAUD is due to limitations outside CCN: either the ARPANET or (more likely) the load on the TENEX system running the WMO.

Table 3 : IP File Transfer Statistics

	TIME OF DAY	RECORD COUNT	REAL BAUD	ELAPSED TIME(sec)	SWAPS	CPU sec	CPU PER MEGABIT
A)	0903	180	1711	22.3	0	0.66	17.3
B)	0911	202	3243	13.0	0	0.74	17.5
C)	0948	202	1338	31.5	3	0.74	17.5
D)	0959	202	3630	11.6	4	0.80	19.0
E)	1019	202	3954	10.6	5	0.84	19.9
F)	1016 (next day)	202	799	52.7	10	0.81	19.2

On the basis of these figures and others, we have concluded that:

* IP file transfers run at 10% to 60% of the Network Baud rate typically experienced with the File Transfer Protocol (FTP) implementations between CCN and a TENEX system.

* When CCN's TSO is lightly loaded, the WMO/IP User under TENEX is limiting the transfer rate.

* when CCN is heavily loaded, the swapping of MP in TSO will presumably slow down file transfers using the present buffering scheme.

The instrumentation in the CCN IP Server is currently inadequate to separate the effect of TSO swapping from the delays introduced by the ARPANET and by the WMO host. However, we have learned enough from these results to design better instrumentation for the File Package. The CCN File Package will be implemented in a manner quite similar to MP. The File Package will execute under TSO like the IP "Files" servers, and in fact some of the code is expected to be the same. File transfer will take place over an MSG "Direct Connection", which will use an Exchange window to a relay process very similar to IPTASK. Thus, Figure 1 with suitable relabeling will be a schematic of file transfers using the CCN File Package.

One significant instrumentation problem is that there is no way at present to determine reliably the elapsed time for an IP "GET FILE" operation, i.e., sending a file from CCN. (The astute reader will observe that all the results given earlier were for SEND FILE operations.) MP instrumentation considers that file transmission is complete when the file, followed by an End-of-file indication, is completely formatted in output buffers. The internal structure of MP includes a separate output buffering process that handles the mechanics of shuffling buffers from output queue to Exchange queue to free queue. MP is configured with 15 buffers of 1K bytes each, shared between input and output. Many short IP files are fully contained in these buffers, so the entire file transmission appears to the MP instrumentation to happen in one flash of computation. As a result, the transmission rates reported by IP for GET FILE operations are always high, and often absurd.

We should note that the output queue handling process within IP is only the first of several processes that handle the messages and may introduce arbitrary buffering and transmission delays; others are IPTASK, the ARPANET, and probably one or more processes within the receiving host.

To improve the accuracy of the transmission rate statistics, one could ask one or more of these processes to report back to MP when the End-of-file passes. The farther towards the receiver this is done, the better the measurement of overall transmission time. Note, however, that a process can recognize End-of-file only by parsing the entire message stream. It is not reasonable to ask any of the intermediate processes to do this; parsing of the stream should be the exclusive job of the ultimate receiver, WMO.

An end-to-end acknowledgment of the End-of-file is thus necessary to allow the donor to reliably measure file transfer rates. The IP protocol includes such an acknowledgment message from the server, but not from the user. To allow symmetric measurements, we propose the following change to the IP protocol:

> After receiving an End-of-file mark terminating a GET FILE operation, the IP User program should "promptly" return a positive acknowledgment message to the Server. File transmission rates will be based upon the elapsed time from sending the first byte of the message to receiving the End-of-file acknowledgment.

The instrumentation of the File Package for performance evaluation and tuning will be considered in greater depth in a future technical report.

2.4. BATCH JOB TABLE RESYNCHRONIZATION IN NSW

2.4.1. OVERVIEW

In the three models of NSW batch processing discussed earlier, the Works Manager Operator (WMO) assigns jobs to IP Servers (or Batch Job Packages or Batch Foremen) operating on Batch Tool-Bearing Hosts (BTBH's). In the IP and the Batch Job Package (BJP) models, the batch job execution sequence (including input file pre-staging) is completely driven by the WMO. In the proposed Batch Foreman (BF) model, on the other hand, once a batch job had been assigned to a BF, the BF would take responsibility for causing its execution.

Under any model of batch operations, the WMO and the BTBH's must agree on "handles" or names for the jobs currently active. The WMO and the host operating system ("HOS") of each BTBH will have tables of all active NSW jobs on that host (although the local HOS table probably includes non-NSW jobs too) In addition, the IP Server/BJP/BF may or may not keep an active-job table. In any case, it is necessary to keep all these active-job tables in synchronism, i.e., containing the same jobs, if NSW is to provide reliable batch service for users. Furthermore, the batch-job handles are used for reporting job charges from the BTBH to the WM. Therefore, NSW accounting reliability depends upon the manner in which the WMO and BF recover from abnormal situations.

Problems arise, for example, if any of the three job-processing sub-systems -- WMO, IP Server/BJP/BF, or BTBH HOS -- crashes and is "cold-started", i.e. if its active-job table is cleared of all entries. A robust NSW batch system should be designed to "automatically" resynchronize all three tables after a cold start of any single system or combination of two systems, or after any other failure causes a discrepancy among these tables.

A serious side effect of synchronization failures is the "abandonment" of batch job files on the BTBH disks. Every batch tool has at least one input file, and many tools create several output files in addition to the primary output ("SYSOUT") file. The input files must be deleted and the output files must be "delivered" back into NSW. Failure to delete files will cause a loss of disk space at the BTBH. Failure to deliver output files to NSW will cause the results of good batch runs to be lost to the user. This is potentially just as serious as the loss of files created by interactive tools, although the effect in the batch case is the loss of processing costs rather than man hours. Clearly, the job table synchronization mechanism which is adopted must be careful to deliver and clean up files at the BTBH.

2.4.2. PRESENT IP RESYNCHRONIZATION

At present, a batch job is known to the WMO by an integer called the "WMO job number" (currently 1-256), and to a BTBH by a character string which it generates called a "job id". In OS/MVT, for example, a job id is an 8-character name called a "jobname"; at CCN the first six characters of this name are constrained to be the "charge number" (directory) under which the job will execute. The responsibility of binding job id's to WMO job numbers and of maintaining these relationships may be centralized in a single process -- either the IP Server or WMO -- or it could be shared between them.

The IP Server design at CCN used a simple strategy intended to avoid job table resynchronization: there is a fixed binding between WMO job numbers and BTBH job id's. The WMO job number is sent to the BTBH in the SUBMIT JOB message, and the CCN server uses this integer to build the last two characters of the job name in a fixed one-to-one manner. As a result, neither the IP Server nor the WMO logically needs a table of the correspondence between job id and job number, although in fact the actual job id is returned to the WMO for later use in naming the primary output file.

The jobnames generated by this mechanism will be unique (as required by OS/MVT) if: (1) the charge number used by the IP Server for running batch jobs at CCN is never used by any other job source, and (2) the WMO never reassigns a job number until it has sent a DELETE JOB message and received an "Job Deleted" reply from all previous uses of that job.

In the IP design, the only association between batch files and the batch jobs is through the WMO's tables. Therefore, since the WMO does not know the names of the files for a job which "appears" at the BTBH, e.g., because the WMO was cold-started, synchronization failures have often left files abandoned on CCN's disks. During periods of intensive WM testing, it has been necessary for CCN personnel to manually delete abandoned files about once a week.

We will now describe possible synchronization failures with the present job-number/job-id correspondence. A detailed analysis of this problem, based upon the possible states of the WMO-IP-HOS system, is contained in Appendix A.

* Job Lost Permanently at BTBH

There are numerous ways in which CCN can "lose" a batch job which the WMO thinks is submitted. For example, the OS/MVT job queues may be cold-started, or the IP Server's table of completed jobs ("JOB DONE table") might be cold started. Note that in the latter case the WMO will have received a charge for the missing job, in an asynchronous "JOB DONE" message -- probably.

The WMO will discover the disappearance of the job as a result of a STATUS QUERY request for the job.

* Job Lost Temporarily at BTBH

It is possible (but not probable) that a CCN system crash will cause a job to "disappear from view" temporarily and reappear later. If the WMO believes that the job is lost and recovers in a rational manner, the result will be the appearance of an "unknown job" later.

If the WMO happens to submit a job with the same job number while the earlier job is out of sight, it is possible for both jobs to be handed to MP by the output spooler. If this happens MP will simply delete the earlier job output and forget about it entirely (probably a sign of embarassment.)

* Job Unknown to the WMO Appears at the BTBH

There are several circumstances which will cause the BTBH to have a job whose number WMO thinks is unused:

(1) the WMO may be "cold started" while jobs are in progress at the BTBH;

(2) the BTBH may crash after submitting a job but before the "Submit Successful" reply is returned to WMO;

(3) a job may be submitted from some source other than MP with the NSW charge number; or

(4) a job which disappeared from view temporarily may reappear.

None of these should happen, of course, but historically most have occurred. During the development of NSW, cold-starting of the WMO has been a recurring event.

The WMO may discover the unexpected job in one of two ways: it may receive an asynchronous JOB DONE message for a job it does not know about, or the WMO may receive an error reply ("Job Refused by TBH") when it tries to submit a new job using the same job number.

Note that each time the IP Server either restarts after a crash or is reset by request of WMO, the server resends JOB DONE messages for all batch jobs awaiting output retrieval from CCN. Thus, the IP Server keeps "reminding" the WMO about an unknown job, until WMO disposes of the job in one way or another.

The failure of a SUBMIT JOB request caused by the WMO's reusing an active job number is a particularly important case. The WMO has three choices:

(1) choose another job number;

(2) wait a few minutes and try again, assuming that eventually the old job will run and a JOB DONE r message will be sent; or

(3) actively try to free the busy job number by cancelling the job.

Of these choices, the first is difficult to implement in the current WMO, and the second (which is currently programmed into the WMO) could force an arbitrary (e.g., 24 hour) delay in submitting a new job. To avoid this possibly large delay, CCN changed the IP Server to issue an OS/MVT "CANCEL" command for the old job when the WMO submits a new job with the same job number. Choice (3) would make this CANCEL the responsibility of the WMO, but it would require a new IP command to CANCEL a submitted job.

Note also that cancelling a job does not immediately purge the job from the system or free its name; CANCEL simply halts the job's use of system resources for any purpose other than exiting the system. All the output created before the moment of cancellation will still be returned to the user, and the charges will be billed to NSW. The delay after issuing CANCEL and until the JOB DONE message is received at the WMO should never exceed a few minutes, however.

When a JOB DONE message for the unexpected active job does arrive the WMO must issue a DELETE JOB for it. The WMO should also record the charges from the JOB DONE message in an "unknown user" account.

* The WMO Deletes an Active Job

It is possible for the WMO to issue a DELETE JOB request for a job which is still active on the BTBH. If the job is complete and awaiting output retrieval, the IP Server will merely delete its primary ("SYSOUT=A") output files (note that the WMO has the responsibility of deleting any other input/output files for the job) and respond "OK". If, however, the job is in one of the OS queues or in execution, the IP Server will simply reply "Not Done" and ignore the DELETE JOB request.

2.4.3. ALTERNATIVE SYNCHRONIZATION DESIGNS

A careful analysis of the preceding section leads to the conclusion that the current IP Server model for resynchronization can be made to work properly, assuming that the WMO implementation pays careful attention to the various problems which were discussed earlier, and is prepared to handle asynchronous JOB DONE messages at unexpected times. The advantage of this approach is that it generally minimizes the complexity of the BTBH software for NSW.

In the present scheme, the WMO must take full responsibility for file cleanup and delivery at the BTBH. This requires that the WMO save its Local Name Directories [6] in a secure manner over WM and WMO crashes, and use recovery mechanisms that parallel those to be defined for interactive Foremen. This essentially means: the WMO must keep track of the "workspaces" at each BTBH.

We will now discuss some possible changes in the job table synchronization for IP. All but the first of these changes are in the direction of the BTBH assuming more responsibility, simplifying the design of the WMO while increasing the complexity of the BTBH code. Therefore, these changes, if adopted, would be more appropriate to a BJP than to an IP Server.

2.4.3.1. Single Handles

The present IP uses both the WMO job number and the BTBH job id as alternative handles for the same batch job. The BTBH code would be simplified and its design would be more flexible if only one handle, the job id, were used. The WMO would store in its table the BTBH job name returned from SUBMIT JOB, and use it as the handle for all further communication about the job.

2.4.3.2. Active Resynchronization

The resynchronization scheme currently in IP is inherently "passive" on the part of the WMO. Fundamentally, it works because the IP Server actively reminds the WMO about any completed but not deleted jobs whenever the server comes up or is reset. Furthermore, it may be assumed that every job in the BTBH will EVENTUALLY reach the "done" state. THerefore, one can guarantee eventual resynchronization.

On the other hand, we could consider an "active" resynchronization mechanism. We would define a BTBH procedure called BFP-SURVEY to scan the BTBH HOS, locate all NSW jobs, and return a list of them to the WMO. This requires that the IP Server/BFP be able to scan the batch queues of its HOS and to recognize NSW jobs from information stored in these queues. These requirements are not problems at CCN, but could conceivably be difficult on some systems.

Whenever the WMO reestablishes contact with BFP for any reason, either a WMO or a BFP problem, the WMO should issue BFP-SURVEY. The resulting list of job names will define the set of NSW jobs at the TBH, and the WMO can compare this against its table of outstanding jobs. The WMO can take corrective action for any discrepancies it detects -- delete unmatched entries from the WMO tables and send BFP-CANCEL messages for unexpected jobs.

An extension of this mechanism can be used for cleaning up the BTBH file space. We can define a BFP-FILESURVEY procedure, that returns to the WMO a list of all NSW batch files at the BTBH. Of course, if the files belong to a job that the WMO does not have in its tables, then the WMO can only delete the files, it cannot deliver them becuase it doesn't know the user. It may be possible for the WMO to encode the user node name into the stem of the file name at the BTBH, so it can map from file names back into users.

2.4.3.3. Job Table in IP Server

The current fixed mapping of WMO job numbers into job names may cause difficulties for WMO during job submission, as discussed earlier. It would be possible for the IP Server (or BJP) to use a dynamic binding between job id and WMO job number and to maintain a local table of this correspondence.

With this change, the following resynchronization algorithm could be used:

2.4.3.3.1. Events at the User (WMO) System:

* STATUS QUERY yields "Job Not Found": the WMO should skip to the file cleanup portion of its job-handling program.

* Unexpected JOB DONE message: WMO should transmit an immediate DELETE JOB message. The JOB DONE message just received is the WMO's billing vehicle and should be directed to the "unknown job" account.

2.4.3.3.2. Events at the IP Server/BJP System:

* DELETE JOB request:

```
If the job number is marked assigned,
   then | if its status is "done",
        |    then delete its output file;
        |    else "cancel" it.
        | Unbind the job number/job id.
        | Mark the job number unassigned.
Return an "OK" response.
```

Notice that the "cancel" will not cause the job to vanish from the HOS, merely cause it to terminate execution prematurely. The HOS will still deliver to BJP all the output created so far. This requires that the binding between a job number and a job id be dynamic -- then, when a job is "cancelled" and its job number is marked unassigned, it will not matter that the HOS does not immediately release the job id. Job charges will still be made available when the job's output is returned to BJP, so they need not be retrieved at cancel time.

* IP SUBMIT JOB request:

If the job number is marked assigned, then simulate DELETE JOB without a reply. In any case, use a random string generator to generate a job id and attempt to submit the job to the HOS using this id. Repeat this process until an acceptable id is found, or until all possible strings have been generated. If an id is found, bind it to the job number and mark the number assigned. Otherwise, return an error response to WMO.

* The HOS passes BJP a job, but its id is not bound to a job number:

Send the usual asynchronous JOB DONE message to WMO, specifying the "unknown" job number.

* BJP initialization finds a job id bound to a job number, but the job id is not found in the HOS and is not "done":

Mark the job "done" and schedule transmission of a JOB DONE message for it. This message should probably be sent in response to the next RESET message received. The charges in the message should be zero.

2.4.3.4. LND Saved at BTBH

We could consider giving the BTBH responsibility for cleaning up its own batch files. To do so, the BJP would need to maintain a table associating files with jobs. The WMO would call a BJP procedure to establish a "job entity" (a "workspace") in the table at the BTBH before files were allocated for the job; the batch job handle would be a handle for this workspace. Subsequent file requests from WMO would need to specify this workspace handle, and the BJP would need to secure the table over system crashes. Using the table, the BJP would clean up its own file-space when it discovered that the WMO no longer knew about a job at the BTBH. This would require the BJP to call DELIVER directly for orphan output files.

2.4.4. THE MSG-BASED BATCH FOREMAN (BF)

Under the Batch Foreman model, full control over a batch job, including its file movements, resides at the local BTBH. As a result, the BF will keep a Local Name Directory ("LND") for the batch job files, and will be required to keep this LND secure over crashes [6]. The BF will furthermore use recovery mechanisms that parallel those to be defined for interactive tool Foremen.

We will not consider the BF model further in this report.

2.5. FUTURE DEVELOPMENT OF THE BATCH SYSTEM

Before leaving the subject of the NSW batch system, we will briefly list the areas in which further development will be desirable.

2.5.1. JOB CANCEL COMMAND

Batch tool jobs, particularly those requiring large machine resources, may be in the BTBH queue for many hours before processing begins. Occasionally the user who submitted the job may later decide to withdraw it without (further) processing. For this purpose, IP needs a CANCEL JOB command.

When the user requests cancellation of his job, the WMO must take the following steps:

* Send the CANCEL JOB message to the BTBH;

* Wait for a JOB DONE message containing the charges accumulated to date for the job;

* Send a DELETE JOB message to the BTBH and wait for positive acknowledgment.

2.5.2. FILE CLEANUP

As discussed earlier, there is a seriou problem of accumulation of "lost" IP files on the CCN disks due to job table synchronization failures. Even after the job table resynchronization problem is solved, morover, system crashes may cause files that do not appear in the WMO tables to be created at the BTBH. Suppose the WMO sends an ENTER FILE NAME message, specifying a partial name, then crashes before receiving the full name from the IP Server. Even if the WMO tables are intact, if the file name is lost in the crash, the WMO can learn about that file only by a direct search procedure, such as the BFP-FILESURVEY procedure suggested earlier.

Ultimately, the "garbage file" problem is tied to the management of the TBH disk space. The WMO currently assumes infinite disk space at the BTBH, but this assumption will clearly fail in a production environment, particularly if output files are left on the BTBH until called for.

An alternative to the FILESURVEY procedure would be a BF-OLD-FILE-PURGE procedure. This procedure would scan the local BTBH file-space and delete all files older than some number of days; this number would be a parameter to the procedure (thus giving the WM control over the purge interval.)

2.5.3. TURNAROUND PREDICTION

A batch tool user should be able to get a statistical prediction of the expected turnaround time on each job that is waiting in a batch TBH queue. We believe that this could be accomplished at CCN by a process that periodically sampled the batch queues and maintained time-weighted averages of jobs and MUS (machine resource measure) serviced per hour. These averages would be kept separately for each batch input queue.

2.5.4. JOB JOURNAL

A typical batch job passes through multiple steps during its processing; for example, it may compile, linkage edit, and finally load and execute. A user normally learns the result of executing each step of his job by inspecting the primary output file. Often this is rather awkward, because traditional batch-processing systems were designed for high-speed printer output rather than terminals. To find the relevant messages buried in the file, the user must know the "magic" strings to be used as search arguments in the text editor used to inspect the file.

An attractive alternative is for the BTBH to extract or create messages containing compact summaries of the job progress and to collect these messages in a "Job Journal" file for each job. The Job Journal might be sent to the user through a "mailbox" mechanism rather than as an explicit NSW file.

2.5.5. OPERATOR MESSAGES

Users sometimes need to send messages to the BTBH computer operator and possibly to receive replies. In a batch-processing environment, this may involve instructions on special tape setup requirements, special forms, or other "administrivia".

NSW should provide a uniform user interface for talking with the operator on any host, particularly any batch host. This probably requires definition and implementation of additional IP messages.

3. INSTALLATION OF BATCH TOOLS

3.1. PACKAGING BATCH TOOLS

An important objective of NSW is to give a uniform appearance to the system environment. For batch tools, this means packaging them in a manner which is clean and easy to understand as well as uniform across hosts. In this section, we will discuss the packaging problems for batch tools on an IBM 360.

* Job Control Language

  The purpose of IBS is to completely hide the details of the job control language of a particular batch host from NSW users. Although some honing of the user interface will probably be desirable, IBS does accomplish this purpose.

* Resource Limits

  One of the difficult problems for a new or inexperienced user of a batch system is choosing appropriate resource limits for his job [10]. The choice of resource limits in many systems affects turnaround time; for example, they may use a shortest-job-first scheduling discipline. In some systems, the resource limits will also affect charges.

  We propose to approximate the required resource limits for compilation steps by simple linear functions of the size of the input files. The coefficients of the formula may depend upon the choice of certain options: for example, the optimization level for a compiler.

  This linear computation is of course only approximate; there are other variables, such as the density of tokens in the program text and its use of complex language features. However, we hypothesize that a practical upper limit on resource use can be computed from linear formulas for a variety of commonly-used programs at CCN. If the formulas only give the binary order-of-magnitude, they are sufficient in many cases.

  The IBS writter for the batch tools currently installed at CCN does use such linear formulas to compute default resouce limits, but allows the user to override these defaults if necessary.

* Environment Errors

In OS/MVT, jobs can fail in a number of ways that depend upon the operating system environment and are reported to the user in an astoundingly obscure manner. The average user may not know immediately that a "213" probably means a file did not exist, a "322" means that CPU limit was exceeded, and "B37" means that there was insufficient disk space to create a new file [11].

Every operating system has its share of obscure messages that come out of the bowels of data management, for example, and whose wording is clear only to a system programmer. When a tool, either batch or interactive, is installed in NSW, it is desirable to replace these utterances with messages couched in terms any NSW user is likely to understand. In the case of OS/MVT batch processing, there is the additional problem that the critical message is likely to be buried in 150 lines of system messages, and the user will have trouble locating it. A solution to this problem, an Output Abstracter, is proposed below.

* Output Selection and Viewing

NSW envisions primarily interactive access to all tools, including batch tools. The user is expected to have a terminal, perhaps a slow terminal, rather than a high-speed line printer. This implies some significant problems in dealing with batch output files in the NSW.

Consider a typical simple job, compiling and executing a Fortran program. There are three processing programs involved (Fortran, the Linking Loader, and the user's module), each of which produces a listing file. If the user runs this job in a standard IBM 360/370 environment, he will receive a printer listing which includes all of these files, one after the other, and also system messages either interspersed or collected at one end. "Reading" such a listing online is a chore even for an expert. Obviously one needs some better tools for handling the output.

One could consider parsing the output stream and transform it into an NLS file. The resulting file might have a standard structure like:

1 : Compiler output

1a: Options

1b: Source listing

1c: Symbols

1d: Errors

2: Loader

2a: Options

2b: Control cards (if any)

2c: Map/cross reference

2d: Errors

3: Execution Step

3a: Parms to execution step (if any)

3b: Execution listing

The machinery of NLS could then form the basis of an output-viewing tool.

We have adopted a much simpler strategy for early NSW use. Each of the different output files (corresponding to the highest-level NLS branch above) is delivered as a distinct NSW file. The user must currently specify the NSW name of each when he describes the job in IBS. (A naming scheme based upon semantic attributes may be used to ease this burden in the future.)

In any case, since each major segment of the job output is in a separate NSW file, the user can view only those segments he needs. For example, if the compilation failed because of syntax errors, he can ignore the loader output file. The system messages are placed in a separate file which the user will need to inspect only if he experiences difficulty with the HOS environment.

3.2. A BATCH OUTPUT ABSTRACTER

From the foregoing discussion, it is clear that NSW users of batch tools at CCN will need a concise message or set of messages summarizing the results of a batch job. One possible mechanism for creating these messages would be through the "encapsulation" of the batch tools; that is, the batch jobs would run in a modified OS/MVT environment in which certain system calls (those used for writing system messages) would be intercepted, interpreted, and used to create corresponding NSW messages. Due to the organization of OS/MVT this interception requires relatively difficult modifications to the base operating system, so we reject this alternative.

We propose instead a program which we will call an "Output Abstractor", or OA, which would execute after the batch job. The OA would parse the primary output file from the batch job, (i.e. the stream of system messages) and abstract a concise summary of job execution.

OA output would include the following kinds of information for each step of the job:

* Success or failure

* Reason for failure

* Files created or not

* Resource usage

Ideally, the OA should have as additional input the tables built by IBS for the WMO for this job. This would allow the OA to use the actual NSW names for the files, for example. One way to accomplish this would be to include the OA within the Works Manager complex, so that the triumvirate IBS-WMO-OA would be involved in every batch execution under NSW. Another approach would be to move the file pre-staging and post-staging functions of the WMO into the Batch Foreman of each BTBH. In this model, the WMO would send the job description tables built by IBS directly to the BF; the latter would call WM-OPEN and WM-DELIVER to initiate the necessary file operations, and complete the JCL skeleton with the actual local file names. The BF could then call OA when a batch job completed, passing OA the table of NSW file names f[illegible] job.

3.3. INSTALLATION PROCEDURE

We will now list the steps which are involved in the installation or "mounting" of a batch tool at CCN. This procedure has been followed for five tools to date. We begin with the tool and its supporting documentation.

* Typically a tool arrives at CCN as a load module library on magnetic tape, although the tool may come as a set of source modules to be compiled/assembled and linkage-edited.

* With the tool itself, we need a set of test cases adequate to give some assurance that the tool is properly installed.

* We need an installation manual which details the resource requirements and installation procedures for the tool.

* We need a user's manual in order to understand the options and develop an NSW installation which is user-oriented.

Given this source material, we can proceed with the initial installation of the tool.

* The first step is to create a working copy of the tool at CCN, by generating or reloading the necessary load modules.

* Using normal OS JCL, we execute the tool with the test cases to make sure it works properly.

* From the tool documentation and from the results of test runs, we determine the resource requirements for the tool:

  -- files (names, attributes, and sizes);

  -- machine resources, (e.g., CPU time, I/O, and region) as a function of the size or other attributes of the input; and

  -- the JCL required to run the tool at CCN.

* We also decide which output files should be delivered separately and which combined into the primary output (as discussed earlier.)

* In general, a cataloged procedure for the tool must be built and placed in the CCN procedure library. (For existing tools, such procedures already exist.) Building as much of the JCL as possible into the cataloged

procedure, rather than including it in the JCL kernel used by IBS, shortens the cycle for later JCL changes.

* From the cataloged procedure design, we determine the JCL kernel that IBS/WMO must produce.

* We determine also the variable parameters which IBS must gather from the user, the linear formulas for determining default resource limits, and the defaults for other parameters.

* All of these requirements are incorporated into a "Tool Installation Specification" document for the tool. This document is passed to the organization responsible for creating the IBS for the tool, and to the tool vendor for approval and verification.

This completes the initial installation of the tool at CCN, and the preparation for NSW installation.

Initial installation of the tool in NSW now occurs.

* The reponsible organization (currently MCA) creates and installs the necessary IBS [Note 3]. as well as the Tool Descriptor.

* The responsible organization writes and installs the NSW HELP files for the tool. (This step is not currently being carried out.)

* Finally, the Tool Vendor is able to test the entire assemblage -- Tool Descriptor, IBS, HELP files, cataloged procedure, and tool -- within NSW. This test may lead to iteration of the earlier steps.

This completes initial tool installation. However, tools are seldom static; updated versions will certainly appear. This may involve little more than installation of a new load module at CCN, and the following steps:

* Rerun the local test using local JCL and the test cases to make sure the new module is properly installed.

* Repeat tool testing within NSW. This requires that the IBS allow access (not necessarily conveniently) to a later (or an earlier) generation of the module.

* When all is verified and the users have been notified, advance generations of the module.

4. AN ANALYSIS OF FILE ATTRIBUTES

In order to accomplish automatic file conversion, the design of the NSW File Package [5] must include the definition of an appropriate set of NSW file attributes. Many CCN tools have strong restrictions on the structural attributes that they can accept in input files. Furthermore, in general the complete structural attributes of a file stored on a particular TBH are not recorded in any file directory on that host and perhaps cannot even be deduced from an examination of the file; however, they often were known to the tool which created the file originally.

To solve these problems, the automatic file conversion mechanism of NSW is planned to operate in the following manner.

* A tool which creates a new file and delivers it to the NSW file catalog will declare at delivery time the values of all relevant attributes of the file.

* When a file is transferred across the ARPANET to a host with a different operating system, the source host will translate it into a canonical format, called "IL" for Intermediate Language. The structural information about the file in the NSW file catalog must be sufficient to allow this translation.

* A tool will "know" the required format for all of its input files; this information will be recorded in the tool descriptor.

* Given the file requirements of the tool and the structural description recorded in the File Catalog, the File Package will attempt to automatically translate an IL encodement of the file into proper format for the tool.

This Section is a working paper which attempts to organize the various file "attributes", "types", "structures", etc., and to discuss their meaning and values. We will make particular reference to files on IBM hosts.

4.1. CLASSIFICATION OF ATTRIBUTES

The attributes of NSW files will be divided into three different categories, as follows:

* GLOBAL ATTRIBUTES

  We will define an attribute to be "global" if all physical copies of an NSW file will have the same value for this attribute, and changing this value would require a non-invertible transformation of the file. Thus, a copy of the file with a changed value of the attribute could not be guaranteed to be logically equivalent to the original. A global attribute is logically associated with the NSW file name, rather than with an individual physical copy.

* PHYSICAL COPY ATTRIBUTE

  A Physical Copy (or "PC") attribute describes some property which may differ among different physical copies of the same file, even though the copies are logically equivalent. That is, changing the value of a PC attribute does do not alter logical data, i.e., involve an invertible transformation.

  Notice that by their definitions global and PC attributes are disjoint, and together the two categories include all possible NSW file attributes, i.e., those attributes which have an NSW-wide meaning.

* LOCAL SYSTEM ATTRIBUTES

  Some hosts (e.g., IBM systems) record their files on disk with a physical format which has meaning only to the local host operating system or to another host of the same family. In addition, some host families may support transfer of files with family-dependent structures, not defined in NSW. Both kinds of attributes, physical format and family copy, will be called "local system" attributes, since they are not NSW-wide.

We will also speak of "Tool Copy" attributes, used by the Tool/Foreman to describe the Tool Copy file which the tool requires. Since Tool Copies are not constrained to be logically equivalent to NSW files, the question of invertability of transformations is not relevant. Hence Tool Copy attributes may fall into the global, PC attributes, or Local System categories.

4.2. OPERATIONS USING ATTRIBUTES

Various NSW operations are controlled by the different file attributes. We will now describe these operations and generally discuss the attributes required, without going into details of the attributes' structures and values. The next major section will organize the attributes that seem necessary, suggesting reasonable structures and values.

4.2.1. NAME DISAMBIGUATION

The Works Manager performs "name disambiguation" to select a particular NSW file from the file catalog [2,6]. The Tool Copy request includes a "filespec" -- a partially specified file name. This is matched against the NSW Filenames using a complex string-matching algorithm. If a suitable match is not found, a new filespec must be obtained from the Front End or from the Foreman.

If the resulting list is empty, the Works Manager returns a failure indication to the Foreman. However, if the list contains more than one file the WM may further match their global attributes against the given set of Tool Copy attributes, selecting the single file which most nearly satisfies the Tool/Foreman's request. If the list of files still contains more than one, the Works Manager must appeal to the user (via direct conversation with the Front End.)

Since the subsequent operation of creating a tool copy is allowed to make non-invertible transformations on the file, the file selected by the name disambiguation process is not required to have identical global attributes to the Tool Copy request. However, certain global attribute conversions are considered impossible (e.g., binary to character), and these should cause rejection by the Works Manager. Otherwise, the File Package called upon later to do an impossible conversion would have to refuse all physical copies of the NSW file; this would effectively include the File Package in the disambiguation loop.

The Tool Copy request may include the following attributes for this match:

* Semantic Type

  The Tool Copy request includes a "semantic type" selected from a broad list of names defined across the NSW. This is matched against the global Semantic Type attribute, which is whatever the file's creator claimed the data to be, and/or against a collection of tool warranties or de-warranties. If a match cannot be found, the Works Manager must obtain a new value for the Tool Copy's semantic type either from the Front End or from the Foreman.

* File organization

  In order to be able to add later support for compound file types (tape reels, IBM PDS's), the Tool Copy request specifies whether the file is to be simple or compound. The Works Manager may block any match which would require a conversion between these two organizations.

* Data Type

  Works Manager may block any match that would require conversion between binary and character types.

* Case

  A tool may require its input file to be "uppercase only". Each NSW file should have a global attribute stating whether its data is intended for use by such tools, and whether automatic conversion to uppercase is to be permitted. If the Works Manager cannot find an NSW file which is in or may be converted to uppercase, it must obtain help from the Front End or Foreman.

4.2.2. PHYSICAL COPY SELECTION

The File Package examines the list of Physical Copy Descriptors (PCD's) for the NSW file selected by the Works Manager, in order to select a Physical Copy. It assumes that the PCD list is not preferentially ordered, making its decision based only on the cost of obtaining a copy. We believe that a File Package will generally have to assume that the costs of all permitted data conversions are similar, and will thus not consider such factors except to block undefined conversions. Hence, the only relevant attribute is:

* Host Number

  The Local File Package will always prefer a Physical Copy resident on the local host if one is available. Otherwise, second choice may be a Physical Copy on a host of the same family, in cases where resource-saving transmission techniques have been worked out for family copies.

  We believe that family copies will be preferred only when the family defines data structures that are not available in File Package IL.

4.2.3. CONVERSION

The receiving File Package is responsible for converting the file into the form specified by the Tool Copy descriptor. Because a Tool Copy need not be logically equivalent to its donor file, File Packages are permitted to perform non-invertible conversions.

In some cases, a File Package may require help from the user to resolve ambiguities of conversion. This means that the File Package should be given the specific MSG address of the Front End, when one exists, and should be permitted to make "help" calls to that process.

Assuming that conversion between binary and character types is forbidden, we can define conversions appropriate for each type.

Conversions will be discussed in more detail later.

4.2.4. PHYSICAL FORMAT DEFINITION

Under a operating system such as OS/MVT, the receiving File Package must define the physical mapping of a file on disk before it can begin writing the file. This implies assigning physical format attributes to the file, including:

* Size

  The receiving file package may need to estimate the total amount of disk space to reserve for the file. This estimation could be based upon one of the following attributes: the donor's allocation page size and number of pages; the record/line size and number of records/lines; the byte size and total number of bytes; or the total number of characters.

* Physical Format

  In MVT, physical format attributes correspond to the DCB parameter in JCL, and include data set organization, record format, logical and physical record lengths, key length and offset, and various option switches. These attributes must be included in the physical format section of the Tool Copy descriptor. Since this descriptor was defined by the local host (or host-family), the local scope of definition of these attributes is not a problem.

4.3. ATTRIBUTE DEFINITION

We can now give a tentative list of file attributes, specifying their structure and values.

4.3.1. GLOBAL ATTRIBUTES

* SEMANTIC TYPE

  The "type" specified by the creator of the file. This type may also be a part of the NSW file name.

* TOOL WARRANTIES

  A list of usage names that have been asserted by tools to be valid uses for the file.

  NOTE: A warrantor is saying, in effect, "There is a possible physical representation of this file which is acceptable for the named usage." For the near future, such warranties will be highly subjective. For instance, most language processors cannot tell the difference between a garbage file and a legitimate source program with some serious syntax error, so these names will have to serve as vague guidelines only. The Works Manager cannot "bond" warrantors. As tools become more sophisticated, we will probably want a "de-warrant" capability.

  While in practice the number of warrranties for an NSW filename may be small, it will frequently exceed one, and could become large.

* FILE ORGANIZATION

  This attribute has the values:

  =SIMPLE

  Virtually all files are of this class.

  =COMPOUND

  Includes constructs like tape reels and IBM Partitioned Data Sets (PDS's). Further attributes take the form of a master set, describing the aggregate, and a list of component sets, describing subfiles. Some form of "override" convention is probably needed here.

This attribute carries a subattribute named NUMBER OF SUBFIELDS.

> NOTE: Compound files will probably not be supported in the near future, but this item is here for completeness.

* KEY TYPE

The type of keys associated with the file.

We believe that NSW should allow one level of keying, to be used either for simple sequence numbers, or for true random access record keys, or for the record numbers associated with random file delivery. Cases where more than one such key type apply to the same file are believed to be rare in theory and nonexistent in practice. The KEY TYPE attribute has the values:

=UNKEYED

The file carries no explicit keys. It may be assumed to be implicitly keyed by a simple 1-origin record count.

=STRING

The keys are arbitrary character strings. The maximum length is carried as a subattribute KEY SIZE.

=NUMERIC

The keys are numeric. It can probably be required that they fit within some standard number of bits. Otherwise, the maximum length would have to be carried as a subattribute KEY SIZE.

> NOTE: Tools reference records of a file by their keys. When the width or precision of the key field changes, the integrity of the values is threatened, because there is no convention that designates which columns of such a field are most significant. Such changes may only be done in making a Tool Copy, and these key attributes are therefore global.

* DATA TYPE

This attribute has the values:

=BINARY

The data is pure binary. The BINARY attribute carries the following subattributes:

* BYTE SIZE

A value in the range 1 - 255. This is the number of bits in a basic data unit -- called either a "byte" or a "word".

* NUMBER OF BYTES

A binary order of magnitude is probably sufficient.

* RECORD STRUCTURE

This attribute has values:

=STREAM

The file is a stream of bytes.

=RECORD

The file has a logical record structure. The attribute carries subattributes:

* RECORD SIZE

This is actually an upper bound. Whether the records are all the same length should not be relevant here.

* NUMBER OF RECORDS

A binary order of magnitude is probably sufficient.

=CHARACTER

The data is represented in a character set compatible with ASCII-68. The attribute carries subattributes:

* NUMBER OF CHARACTERS

A binary order of magnitude is probably sufficient.

* DIMENSIONALITY

This attribute specifies the formatting dimensionality of the data at the time it was delivered/imported into NSW filespace. It has the values:

=1 -- The data is a linear stream of characters.

=2 -- The data consists of characters organized into lines.

=3 -- The data consists of lines organized into printed pages.

=4 -- The data consists of lines with overprinting (backspaces or "skip zeros"), optionally organized into printer pages.

NOTE: We consider the original dimensionality of the file to be an important and permanent property of all physical copies, and therefore it is listed as a global attribute.

Notice, however, that conversions among 1-, 2-, and 3-dimensional data COULD be considered invertible, provided values for the PAGE DEPTH and LINE LENGTH attributes are defined and preserved. We therefore define the "current dimensionality" as a PC attribute (REDIMENSIONED; see below).

Note also that 4-dimensional data cannot be converted to any other dimensionality without losing logical equivalence, although this should be permitted while making a tool copy.

* LINE LENGTH

This is actually an upper bound. Whether the lines are actually all the same length is not relevant here.

* NUMBER OF LINES

A binary order of magnitude is probably sufficient.

* PAGE DEPTH

This information may be supplied by the file originator.

* CASE

This attribute designates whether the file is intended for processing by "uppercase-only" tools, and also whether a File Package may perform an uppercase translation on the data. The interesting cases are:

=UPPER

The data is intended to be uppercase. Any lowercase characters found may be changed to uppercase without altering the logical file contents.

=LOWER

The data is intended to contain both lower and upper case characters. It is not intended as input to tools which only understand uppercase.

=VIRTUAL UPPER

The data is intended for input to tools which only understand uppercase. However, there may exist significant lowercase characters, so the File Package may not arbitrarily force uppercase. The file creator has taken responsibility for this.

NOTE: CASE is a global attribute because translations that alter it are not invertible, and thus, such translations may only be made while making a Tool Copy.

The UPPER value of this attribute describes the logical data. It does not say whether any lowercase characters actually exist in the data, which would describe a data characteristic that may be changed without altering logical data, and which is thus a PC attribute (see the UPPERCASED attribute below.)

The VIRTUAL UPPER value of this attribute will handle the important case of langauge processors which deal with uppercase-only language syntax in which is embedded character-constant data in upper and lowercase. This is a common problem in IBM programming systems.

* TAB DEFINITION

The tab definition consists of two identically structured descriptors, one for horizontal tabs and one for vertical tabs. Each descriptor consists of a count of defined tab operators and, for each operator, its character value and either a list of stops or a uniform increment value. Stops and increments are all specified relative to the non-existent column/line 0.

NOTE: Only at Tool Copy time can it be known whether a tool wishes to process its own tabs. Even then it may not be known whether the tool considers tabs merely as a data compression mechanism or as control data. Certainly nothing precludes the mounting of tab-manipulating tools. For these reasons, it appears that tabs must not be expanded or recalculated except at Tool Copy time. It follows that the tab structure of all physical copies of a file must be the same, and that the tab descriptors are rightly global attributes.

* NUMBER OF CHARACTERS

A binary order of magnitude is probably sufficient.

4.3.2. PHYSICAL COPY ATTRIBUTES

This section could be taken to be a model of the actual data structure of a Physical Copy Descriptor (PCD), although no data representation is meant to be implied.

* HOST NUMBER

  This identifies the location of the physical copy. In the context of a given File Package, it defines a virtual attribute of HOST RELATION, with values of LOCAL, FAMILY, and FOREIGN.

* UPPERCASED

  This Boolean attribute is meaningful only for NSW files with the global attribute CASE=UPPER and means that there are no lowercase characters in this Physical Copy.

* REDIMENSIONED= 1 | 2 | 3

  This attribute, meaningful only if the global DIMENSIONALITY 0 attribute is 1, 2, or 3, gives the actual dimensionality of this physical copy.

  NOTE: If a Physical Copy is made in a dimensionality different from that of the donor copy, the global attributes LINE LENGTH and PAGE DEPTH of the file must be used if given. Whenever either of these attributes was both relevant and null, the copying File Package should assign the value(s) that it intends to use, and so update the Works Manager's data base. File Packages would need the capability to make such updates. Notice that in this case, the values of these data are not critical, so long as the same value is used by all processors of the file. File Packages would need the capability to

  Interpretation of this attribute depends on the location of the file being examined with relation to the examiner. Its values are:

=FP IL

This value represents FP Intermediate Language. Remote files on a non-family host always appear to be encoded in this form, regardless of the value of this attribute.

=FAMILY IL

Within some families, unless they are in File Package IL, remote files on a family host always appear to be encoded in their private IL, regardless of the value of this attribute.

=LOCAL ENCODEMENT

This is meaningful only for locally resident copies. It refers the examiner to the physical format attributes.

* SIZE

This attribute is intended to aid receiving hosts which must pre-allocate space for a new file copy.

* PAGE SIZE

Here "page" really means the natural unit for storage allocation in the local system. On IBM systems, this will usually be a disk track.

* NUMBER OF PAGES

Unlike attributes such as NUMBER OF LINES and NUMBER OF BYTES, these storage units are intended to be those actually used for local space accounting. Therefore, this value may be expected to be fairly accurate.

4.3.3. LOCAL SYSTEM ATTRIBUTES

* HOST FAMILY ATTRIBUTES

This is a free string as far as NSW specifications are concerned. Each host family should agree on attributes and syntax.

* PHYSICAL FORMAT ATTRIBUTES

This is a free string so far as NSW specifications are concerned. Each host may define its own attributes and syntax. For an IBM system, the local host will need to store a lot of information here.

4.4. TOOL COPY CONVERSION

This section discusses briefly the conversions to be performed by the File Package when it creates a Tool Copy. The File Package must consider the global and PC attributes of the chosen physical copy (called the "donor attributes" here), as well as the target Tool Copy attributes.

We first discuss the conversion rules in general. Then, the next section describes in some detail the possible contents of a Tool Copy descriptor for an IBM OS/MVT system.

Many of the conversions are concerned with changing the "shape" of the file or its contents. Changes of shape must be controlled by switches, which we can conveniently summarize with a "TAP indicator". This indicator is a series of three switches with meanings:

T -- a switch permitting truncation when putting a larger entity into a smaller one; if "T" is off, the larger entity is to be "folded".

A -- a switch requiring alignment of each input entity on an output entity boundary.

P -- a switch requiring padding of all entities to the same length.

Attributes that define legitimate conversions include:

4.4.1. Conversion of Keys

The Tool Copy descriptor should specify whether the file is to be explicitly keyed, the type of key expected, its position within the data record, and its sort requirements. The Donor file carries similar attributes, and the File Package must effect a reasonable conversion where indicated. A simple example of this conversion is restoring sequence numbers to columns 73-80 of a card image file. More complicated examples are: sorting a file which was delivered randomly by key and creating an Indexed Sequential file.

4.4.2. Conversion of Binary Data

* Byte Size

  The donor file carries a BYTE SIZE attribute; the Tool Copy descriptor carries BYTE SIZE and a TAP indicator. If the byte sizes differ and the "A" switch is on, each byte of the donor file will be aligned on a byte boundary for the Tool Copy; "P" is assumed. The "T" switch controls whether long bytes are truncated or folded.

* Record Structure and Record Size

The donor file will carry a RECORD STRUCTURE and (if RECORD) a RECORD SIZE attribute. The Tool Copy descriptor will carry the same attributes as well as a TAP indicator. For conversion of RECORD-to-RECORD structure, the "A" switch is assumed, i.e., donor records are mapped into Tool Copy records.

4.4.3. Conversion of Character Data

* Dimensionality

The Tool Copy descriptor should state the range of dimensionalities acceptable to the tool, i.e., whether the tool is prepared to handle lines, pagination, and/or overprinting, or whether the program is only interested in the data as a stream. The donor file attributes specify the dimensionality of the file. The File Package must effect the necessary conversions; conversion from dimension 4 to a lower dimensionality can be defined only be local conventions.

* Line Length and Page Depth

Both the donor file and the Tool Copy descriptor may carry values for these attributes; in addition, the Tool Copy descriptor will include a TAP indicator to handle mismatches.

* Case

If the Tool Copy descriptor requests uppercase but the donor file has case attribute "VIRTUAL UPPER", then no case transformation is performed. If an uppercase file is required and the donor file is "UPPER" and the PC is not "UPPERCASED", then the File Package must perform uppercase translation. If the donor file has been declared with the attribute "LOWER" to prevent uppercase translation, but the tool requires an uppercase file, then the File Package should ask the user if he wishes to force uppercase.

* Tab Expansion

If the Tool Copy descriptor does not request that tabs be left unexpanded, the File Package must expand them according to the tab definition attributes of the donor file. This request might be separately stated for horizontal and vertical tabs.

4.5. TOOL COPY ATTRIBUTES FOR IBM SYSTEMS

We now describe in some detail a set of possible Tool Copy attributes, with special reference to the OS/MVT system at CCN. This list of data could be taken as a model for the actual data structures of the Tool Copy descriptor.

4.5.1. LOCAL NAME=string

The name given the tool copy by the creating File Package.

4.5.2. SEMANTIC TYPE=string

4.5.3. FILE ORGANIZATION = SIMPLE | COMPOUND

4.5.4. KEYS = REQUIRED | GENERATE | FIELD | OPTIONAL | NONE

* REQUIRED -- The tool references file records via previously-bound keys. Thus, the Tool Copy must contain keys obtained from the donor file.

* GENERATE -- The tool needs monotonic keys in the file records; if the NSW file has no keys, then the File Package must generate them.

* FIELD -- The tool reserves a field for keys, but it doesn't matter whether there is any information there or not.

* OPTIONAL -- The tool can process a key field if the donor file provides one; otherwise, the field can be omitted.

* NONE -- The tool makes no provision for a key field. If the donor file includes this information, it is to be discarded.

Unless KEYS=NONE is specified, the following subattributes will be included:

* KEY TYPE = STRING | NUMBER

* KEY POSITION = integer

* KEY LENGTH = integer

These values give the starting position and length of the field. The key is to be inserted into the data record without overlaying any data. The key attributes carry a TAP indicator.

4.5.5. DATA TYPE = CHARACTER | BINARY

=BINARY

If the tool copy is to be binary-encoded, only a BINARY NSW file can be used as a donor. This attribute carries subattributes BYTE SIZE, RECORD STRUCTURE, and for RECORD, RECORD SIZE. BYTE SIZE and RECORD SIZE have associated TAP indicators.

=CHARACTER

If the tool copy is to be character-encoded, only a CHARACTER NSW file can be used as a donor. On an IBM host, conversion between ASCII-68 and EBCDIC will be automatic. Some subattributes are defined:

* DIMENSIONALITIES= integer, integer, ...

  This is a list of the dimensionalities that the tool is prepared to process. In an IBM implementation, the values may be:

  1 -- the file is a pure character stream. This would be recorded as umblocked data (i.e., "F" or "V") on disk. No keys may be used.

  2 -- the file is organized into logical records without ASA carriage control codes (i.e., "FB" or "VB".)

  3 -- the file is organized into logical records with ASA carriage control codes; however, overprinting does not occur.

  4 -- the file is organized into logical records with all ASA carriage control codes permitted.

* LINE LENGTH = integer

* PAGE DEPTH = integer

* UPPERCASE = YES | NO

  The tool does not understand lowercase. The file package must ensure that the tool copy is acceptable.

  * If the donor has the PC attribute UPPERCASED or global attribute VIRTUAL UPPER, no action is required.

* If the donor has the global attribute CASE=UPPER but does not have the PC attribute UPPERCASED, then the File Package must "uppercase" the file during the copy operation.

* EXPAND TABS = YES | NO

  If YES, the File Package is to expand horizontal and vertical tabs while making the tool copy.

4.5.6. Physical Format Attributes

These attributes relate to a Physical Copy and are not intended to be meaningful except to the local host, assumed here to be an IBM OS/MVT system. It is possible that family copy techniques may extend the knowledge of some of these attributes across families, but this is yet to be explored. Some of these attributes are redundant in the sense that the Foreman may use them to derive related global attributes.

* DATA SET ORGANIZATION

  CCN will probably support Sequential and Direct files immediately. Partitioned organization will come later, and Indexed organization will probably have to come eventually.

* RECORD FORMAT

  This includes the values FIXED, VARIABLE, and UNDEFINED. It will determine "P" of TAP for the RECORD SIZE or LINE LENGTH attribute of the Tool Copy.

* LOGICAL RECORD SIZE

  This is the maximum length of a logical record in bytes. It will determine the RECORD SIZE or LINE LENGTH attribute of the Tool Copy.

* BLOCK SIZE

  This is the maximum length of a physical block in bytes.

* RELATIVE KEY POSITION, KEY LENGTH

  This is used for data sets of indexed organization, for those of direct organization which are to be accessed through key search techniques, and for sequential files which contain sequence numbers. It will determine the KEY POSITION and KEY LENGTH subattributes of the KEYS attribute.

5. APPENDIX A -- JOB TABLE RESYNCHRONIZATION ANALYSIS

This Appendix is an attempt to analyze all the possible problems of synchronizing the WMO job number and the BTBH job id under the present IP Server model. This analysis uses a finite-state-machine description of the problem.

5.1. TABLES AND STATES

NSW jobs may be recorded in any of five different tables at the WMO and at CCN. These tables are:

W: the WMO'S 'assigned this BTBH' table.

T: the CCN TMT (hence, the MVT JOBQUEUE.)

C: the MVT Catalog (by cataloging the output dataset name.)

G: the selected GMF queue, used to notify MP of the output.

M: MP's own Job-Done table (stored in the JOBFILE dataset.)

In principle, a batch job could be present or absent in each of these tables independently at a given moment; there are thus 2**5 = 32 possible situations or "states" to the WMO/BTBH system. We describe a state by listing the letters of the tables in which the job appears.

The NORMAL sequence of states for a particular job is as follows:

free: initially.

W: on successful negotiation of WMFINDTBH.

WT: on successful negotiation of IP SUBMIT.

WTC: when the output writer SPOOL3 creates the the output dataset.

WC: when SPOOL3 deletes the job from OS.

WCG: when SPOOL3 enqueues the job in GMF.

WCGM: when the GMF notification is successfully received by MP.

WCM: when MP deletes the GMF message.

WM: when the WMO deletes the output.

W: when WMO negotiates IP JOB DELETE.

free: when WMO pleases.

5.2. PATHOLOGICAL STATE TRANSFERS

However, error exits can occur for each state. The ones that seem probable enough to need attention are:

* W --> W

  if IP SUBMIT fails. This is WMO's problem.

* W --> free

  if WMO is cold-started at this moment.

* WT --> W

  if OS is cold-started.

* WT --> T

  if WMO is cold-started.

* WT --> free

  if both OS and WMO are cold-started.

* WTC --> WC

  if GMF is down while the output spooler SPOOL3 is operating on the results of the job. (NOTE: SPOOL3 has been corrected to defer output processing if GMF is down, so this case is no longer probable.)

* WTC --> TC:

  if WMO is cold-started. This will be detected either as a SUBMIT failure or when things progress to state CM.

* WTC --> C:

  This requires a compound error.

* WC cul-de-sac:

if OS crashes during this state. Theoretically, this state is entered in a "critical section" of SPOOL3, but this is not really possible. It could be argued that the order of things in SPOOL3 should produce a momentary state of WTCG here; that is NOT what happens, though.

* WCG --> WC:

  if SPOOL3 posts GMF when MP is down and then GMF is cold-started when it is up simultaneously with MP.

* WCG --> C:

  This requires a compound error.

* WCGM arrested state:

  if MP goes down. Theoretically, this is a momentary "critical section" state, but this cannot really be enforced. If MP finds this state pre-existing, it will appear to be a duplicate job.

* WCM --> WC:

  if MP goes down and loses its JOBFILE data set.

* WCM --> CM:

  if WMO is cold-started.

* WCM --> C:

  This requires a compound error.

* WM --> W:

  if MP goes down and loses its JOBFILE data set.

* WM --> M:

  if WMO is cold-started.

* WM --> free:

  This requires a compound error.

* free --> T:

  if a local CCN job is submitted with an NSW-like name. This may cause SUBMIT failures. It also could propagate to state CM or M, at which point a spurious message would be sent to WMO. Notice that MP's keeping a complete job table does not help detect this situation.

* free --> G :

  if a local GMF message is sent to an NSW queue. This may cause SUBMIT failures. It also might cause MP to ABEND if the message contained real garbage.

* Any state loses its 'C':

  This means merely that a data set has been scratched by someone or something. The job's OUTPUT is lost, but the jobname-space integrity is not damaged unless MP has to resort to a catalog search to correct other errors.

5.3. RECOVERY FROM PATHOLOGICAL STATES

* W OR WC:

  CCN has completely lost a job that WMO is still aware of. Currently, this won't be detected until WMO sends a STATUS QUERY, to which CCN must reply "Job Not Found". WMO should respond by skipping forward to its cleanup phase, requesting DELETE FILE for all files associated with the job. Otherwise, the files may stay on CCN disks forever.

  However, if MP DOES keep a table of submitted jobs, lost jobs can be identified at MP re-initialization time. Whenever MP connects or re-connects to GMF, any GMF messages are moved into the MP JOBFILE. When it is known that GMF is empty, a search can be made for MP table entries marked 'TMT'. If they are not found, and provided GMF remains empty throughout the search, they can be re-marked as "lost".

  In either case, once it is known that a job is lost, MP can attempt a "rescue" by searching the catalog for all matching DSNAME's, selecting the one last allocated and faking a charge record of zeros. The output thus found might not be the right one, and charging would certainly be fouled up. It would be better to forget the job and let WMO attempt a cleanup.

* T OR TC:

  Either WMO has lost a job or one has been entered locally. If MP keeps his own table, he can distinguish these cases, although it is not clear that this would help. The problem may be found when WMO tries to re-use the job number via SUBMIT JOB request, or it may lie dormant until state CM. The latter case is discussed below. In the former, it is desirable to purge the system of the OLDER job. If WMO lost it, it cannot be cleaned up. If a local user submitted it, it is unauthorized. In either case, the job shouldn't be allowed to use any more CCN resources. However, there is no way for WMO to prevent this.

NOTE: MP has been changed to issue an OS CANCEL whenever SUBMIT JOB fails for any reason. If the failure was not because of a duplicate job name, the CANCEL will fail harmlessly.

In the present system, MP does not keep a table of submitted jobs, and nothing can cause the job number to be immediately freed. WMO would still have to poll it occasionally. It will either go away by itself, in which case a STATUS QUERY message will eventually re-synchronize the two systems, or it will eventually reach state CM, discussed below. However, if MP kept a table of submitted jobs, the mapping of WMO job numbers into OS job names would be MP's responsibility. Whenever it was desirable to make a job vanish, MP could simply remove it from his tables, assigning a new job name to that WMO job number. Assuming the old job was cancelled, it would simply be discarded when it reached the transition to state CM.

* C or free:

  This takes two errors, at least. Neither WMO nor MP know anything about the job, unless MP has kept a table of submitted jobs. In the latter case, MP can delete the output file, but the input files cannot be cleaned up.

* M or CM:

  MP is holding an output job that WMO knows nothing about. It is desirable to delete the job, purge its output file, and ignore any other disk space it may have left allocated. There are two ways to do this:

  * If MP discovers the situation through a SUBMIT failure, MP can simulate DELETE JOB and attempt SUBMIT again immediately.

  * If WMO discovers the situation when he receives a surprising JOB DONE message, he can transmit a DELETE JOB. Note that this may occur after the WMO has sent a SUBMIT message but before the reply.

* WCGM:

  This state will look to MP's NOTIFY function like an illegitimate CM followed by a legitimate WCG. Unless a check is made, he might purge the output but keep the job. The solution is to compare the output dataset names; if they are the same, discard the NEW job, else discard the OLD one.

5.4. RECOMMENDATIONS

* WMO should interpret a SUBMIT failure with code "Job Refused" as a job-table integrity problem. WMO should set a state that will cause periodic polling of the BTBH for the job number while listening for a JOB DONE message. This state should prevail until either "Job Not Found" is returned from STATUS QUERY, or until WMO issues DELETE JOB in response to a JOB DONE message.

* Whenever WMO receives evidence that a BTBH has lost a job, it should skip forward to the data set cleanup portion of its job-handling program.

* SPOOL3 should be changed to pass through state WTCG instead of WC. However, this may require a basic reorganization of SPOOL3 logic.

6. APPENDIX B -- MONITORING IP SERVERS AT CCN

6.1. HOW TO MONITOR

CCN operates IP Servers for NSW systems on both BBNB and ISIC. These servers write detailed transaction logs which can be monitored by other TSO users:

* NW3 monitors all servers for BBNB.

* NW4 monitors all servers for ISIC.

To view one of these monitor streams, first gain access to CCN TSO. Then enter:

```
LOGON NWx/password
```

replacing 'x' with '3' or '4', depending on whom you wish to monitor. TSO will shuffle about a bit and eventually come up:

```
READY:
```

If you are communicating with CCN either through a real terminal that has a locking keyboard or through Server Telnet, you have to lock the keyboard before TSO will display the monitoring data. You can do this by typing:

```
LOCK
```

To unlock the keyboard at the conclusion of monitoring, you must transmit an 'attention' (real terminal) or Control-C (Network Virtual Terminal). If all else fails, simply break the connection to CCN. If you can return to "READY:" state, however, you can leave TSO gracefully by typing:

```
LOGOFF
```

One of the properties of IBM TSO is that a given USERID can be logged onto TSO only once at a given moment. If your LOGON is refused with a message to the effect that the USERID is in use, it means that someone else is monitoring. Come back later.

6.2. MONITOR OUTPUT

The output lines from the servers' logs look like:

```
NWx:  :NSWnnn yymmddhhmmss data
```

where:

* 'x' is '1' to 'A', giving the full userid of the IP Server session being monitored.

* 'yymmddhhmmss' is a date-and-time stamp.

* 'nnn' is a message type code, determining just what 'data' means.

    000: data is a statistical record.

    001: data is a comment record, and should be self-explanatory.

    002: data is an alarm message -- the server is terminating abruptly.

    003: data is a TSO command being issued by the server.

    004: data is the return code from a TSO command. Normally, zero is what one expects.

    005: data is an incoming IP message. The 'handle' and 'data length' fields of the message are binary, so they will usually appear as ':'.

    006: data is an outgoing IP message. There are binary fields as in nnn=005.

    007: unused.

    008: data is an incoming IP data record being transmitted in binary mode. Most of the record will appear as ':'s.

    009: unused.

    010: data is an outgoing IP data record being transmitted in binary mode. Most of the record will appear as ':'s.

    011: data is an operator message.

7. REFERENCES AND NOTES

7.1. References:

[1] "The National Software Works," Robert E. Millstein. Document No. CADD-7607-3011, Massachusetts Computer Associates, Wakefield, Mass., August 1976.

[2] "Works Manager Procedures". Chapter 2 in "Semi-Annual Technical Report," Document No. CADD-7603-0411, Massachusetts Computer Associates, Wakefield, Mass., March 1976.

[3] "An IP Server for NSW-- Semiannual Technical Report," R. T. Braden and H. C. Ludlam. CCN Technical Report TR-8, UCLA Campus Computing Network, UCLA, April 1976.

[4] "MSG: The Interprocess Communication Facility for the National Software Works," NSW protocol Committee. Document No. CADD-7612-2411, Massachusetts Computer Associates, Wakefield, Mass., December 1976 (rev.).

[5] "File Package: The File Handling Facility for the National Software Works," Paul M Cashman, Ross A. Faneuf, and Charles A. Muntz. Document No. CADD-7612-2711, Massachusetts Computer Associates, Wakefield, Mass., December 1976 (rev.).

[6] "The Foreman: Providing the Program Execution Environment for the National Software Works," Richard E. Schantz and Robert E. Millstein. Document No. CADD-7604-0111, Massachusetts Computer Associates, Wakefield, Mass., March 1976.

[7] "Programmers' Guide to the Exchange," R. T. Braden and S. C. Feigin. CCN Technical Report TR-5, UCLA Campus Computing Network, UCLA, March 1971.

[8] "MP Execution Parameters". In Appendix C of Reference 3, pp. 82-84.

[9] "A Protocol for Packet Network Intercommunication," Vinton G. Cerf and Robert E. Kahn. IEEE Transactions on Communications, Vol COM-22, No. 5, May 1974.

[10] "SUPP.JCL.ESTIMATES". CCN User Document S043, UCLA Campus Computing Network, UCLA.

[11] "BASIC.DEBUGGING". CCN User Document B045, UCLA Campus Computing Network, UCLA.

7.2. Notes:

Note 1:

In order to understand the difficulty of file translation, it is helpful to realize that host operating systems are broadly divisible into two major types: "unit-record" (or "punched-card") and "paper tape" systems. This terminology reflects the dominant input/output medium during the early development of each computer system. This simple mechanical "happenstance" has had a profound effect on the ultimate system software development.

"Unit-record" systems generally started with punched card input and line printer output; IBM systems fall into this category. These systems consider a file to be divided into groups called "records". Records are delimited by implicit control information, i.e. information which is not part of the data. Print files may use ASA Carriage Control characters at the beginning of each record, or may store the print format control implicitly.

"Paper-tape" systems generally developed in environments where paper-tape was used for I/O. The PDP-10 falls into this category. A file on a paper-tape machine is generally conceived to be just an undivided stream of characters. Logical divisions of this stream (e.g., into lines) are imposed only by the particular program processing that file. This division occurs through interpretation of control characters (or character sequences) included explicitly as data in the file. In particular, end-of-line and printer format information is included within the stream as control characters ("format effectors") whose interpretation was established originally by the design of a Teletype machine. Unfortunately, so simple a thing as the end-of-line character (sequence) differs across systems even on the same host; different PDP-10 systems use CR, LF, CR-LF, and US, for example.

In order to transfer a file from a "paper-tape" host to a "unit-record" host, the stream must be scanned character-by-character for the control characters which must be translated into implicit end-of-record markers and ASA Carriage Control characters before the receiver records the data on disk. Since the interpretation of even the universally-understood control characters differs from one "paper-tape" system to another, this scanning must generally be performed by the "paper-tape" host. In NSW, the result of the scan is a universal "unit-record" format called "IL", which each "unit-record" host can map into his file system. For the reverse translation, "unit-record" to "paper-tape", the same considerations hold.

The end-of-line marker is relatively simple compared to the control functions HT (horizontal TAB), VT (vertical tab), BS (back space), or CR (isolated Carriage Return). The "unit-record" systems generally will accept none of these "format effector" control characters embedded within text.

For the "unit-record" systems which use explicit ASA Carriage Control characters in a print file, a print file is significantly different from other files. Most IBM compilers will not accept a print file as input, for example. Therefore the rules used for translation from "paper-tape" to "unit-record" must depend upon the semantic content of the file. A Fortran source file at CCN had better not have carriage control, for example, or the IBM Fortran compiler will produce useless results.

Note 2:

The term "directory" has a well-established meaning in modern operating system design. We use the term in this document with particular reference to the "login directory" of BBN's TENEX system. The TENEX directory carries the password controlling login and file access, and its name forms the first-level of file pathnames. At CCN, the first-level of a full file name is the catenation: "Account.Userid"; this same pairing also carries the TSO logon password. Hence, it is natural to speak of the pair (Account, Userid) as the TSO "logon directory". Similarly, in batch processing at CCN, the account number alone carries the password. Hence, we speak of the account number as the name of a "batch directory".

Note 3:

MCA plans to make IBS table-driven, easing the difficulty of installing new tools.